METHOD AND APPARATUS FOR IMPROVING YIELD BY DECOMMISSIONING OPTIONAL UNITS ON A CPU DUE TO MANUFACTURING DEFECTS

FIELD OF THE INVENTION

[0001] The invention pertains to the field of design and packaging of large, complex, integrated circuits such as processors. In particular, the invention relates to a method of design and packaging CPU integrated circuits so that partially-defective processor circuits may be sold as reduced-performance processor circuits.

BACKGROUND OF THE INVENTION

[0002] It is well known in the art of integrated circuits that manufacturing processes are imperfect – on each wafer some, but not all, integrated circuits function fully. It is also known that defects tend to occur in clusters. Therefore, for circuits having multiple, large, functional units, there will be a substantial population of manufactured integrated circuits where one functional unit is defective, or even a small portion of a functional unit is defective, but other functional units on the same integrated circuit function properly.

[0003] The probability that an integrated circuit will have one or more defects increases as the size of the integrated circuit increases. Further, the cost of fabrication increases as integrated circuit size increases. A high performance single or multiple-processor integrated circuit can be quite large. It is therefore desirable to find ways of selling at least some of those integrated circuits that contain one or a few defects.

[0004] Typically, integrated circuits are tested before they are packaged. Those processor circuits that have defective units are often discarded before packaging; their fabrication cost is wasted but further investment in them is prevented.

[0005] Solutions that permit selling partially-defective integrated circuits are known for memory integrated circuits. For example, memory integrated circuits often have spare blocks of memory that can be substituted for defective sections of the circuit. Before spare blocks became common, memory circuits were occasionally packaged with a high-order address bit wirebonded to a constant, then sold as memories of half capacity, with defective sections of the chip disabled. Wirebonding a bondpad to a constant to configure an integrated circuit is known as a bonding option; bonding options may also be implemented through a package trace where a particular package lead is tied to a

particular logic level. Bonding the high-order address bit to power or ground permitted sale of packaged circuits having partial capacity, with the same circuit pinout for defects in either the high or low half of the memory array.

[0006] Modern integrated processor circuits of high performance are fabricated with at least some cache memory on the processor integrated circuit. Some of these circuits have been designed with bonding options such that a portion of cache may be disabled; a technique that permits product differentiation as well as sale of partially defective circuits. Some of these circuits also have spare blocks of memory that can be substituted for defective sections of cache.

[0007] It is known that some available processors have multiples of certain functional units. For example, a processor integrated circuit may have more than one integer execution unit, or more than one floating point execution unit. These units are referenced herein as redundant units. Historically, processors with one defective integer unit, or one defective floating point unit, are typically discarded. Similarly, if multiple processor integrated circuits are sold with a defect in an integer unit or floating point unit of one processor, the entire defective processor is disabled.

[0008] It is also known that some processors have functional elements that enhance performance but which are not essential to processor operation. For example, deletion of a branch prediction unit, speculative prefetch unit, or a speculative execution unit may degrade but not kill performance. These functional elements are referenced herein as performance-enhancing units.

[0009] Branch prediction and speculative execution are common techniques for minimizing the impact of conditional branch instructions on processor performance.

[0010] Branch prediction involves using dedicated hardware to guess whether conditional branches will be taken or not taken. This may be done by tracking recent results of particular conditional branch instructions, and predicting that a particular branch instruction will cause a branch if it caused a branch when it was last executed.

[0011] Speculative execution involves a machine's beginning execution of instructions after a conditional branch is encountered in an instruction stream when it is not yet known if the instructions are among those that should be executed. Typically, speculatively executed instruction results are not made permanent until it is determined whether they should have been executed. If instructions are speculatively executed, and it is determined that they should not have been executed, their results are discarded.

[0012] Integrated circuit fabrication processes are known that are capable of producing read-only memory cells that can be programmed after wafer fabrication is complete. These processes have been used to create programmable memory devices, programmable logic devices, and other circuits. Processes of this type that use fusible links or laser-burnable links are known. Other processes are known that create programmable cells that are programmed by tunneling charge onto a charge-trapping layer, such as a floating polysilicon gate or an oxide-nitride boundary, in the gate region of a device.

[0013] It would be desirable to recover some revenue by using post-fabrication programmable cells to selectively disable defective functional units on processors that have multiples of functional units of those types, and selling the partially defective devices as lower performance processors.

[0014] It is also known that different application programs have different processor requirements. For example, some programs have much greater floating point computational requirements than others. Similarly, the percentage of branch instructions may also vary significantly.

[0015] Built In Self-Test (BIST) is a design technique wherein circuitry of a functional unit is designed, and some additional circuitry added to the unit, such that the unit can effectively test itself.

SUMMARY OF THE INVENTION

[0016] A high performance processor integrated circuit has multiple functional units of various types, including integer execution units, and floating point execution units. The processor is designed and fabricated with programmable cells that may be programmed after wafer fabrication is complete.

[0017] When the processor integrated circuit is tested it may be found that one functional unit of a particular type is defective, while one or more functional units of the same type is fully operational. When this occurs, a programmable cell is written to disable the defective unit and reroute all operation needing a unit of that type to the fully operational units of the same type. This is done by programming appropriate bits in a resource status register, thereby allowing the instruction decode and dispatch units of the processor to dispatch operations only to the functional unit or units, while stalling as necessary when more units of a given type are requested than are available.

[0018] When the processor integrated circuit is tested it may be also be found that a performance-enhancing element, such as a portion of cache or a branch prediction unit, is defective; while sufficient other units of the circuit function that it may serve as a functional processor of reduced performance. Integrated circuits having these characteristics also have appropriate bits of the resource status register programmed to disable the defective units.

[0019] The characteristics of each processor integrated circuit are thereupon compared to a relative performance table. The relative performance table relates defective, disabled, functional units of each integrated circuit to a relative performance level. Those devices having particular performance levels are grouped together and marketed at a lower price than fully functional devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[0020] Figure 1 is a block diagram of portions of a prior art processor;

[0021] Figure 2, a block diagram of portions of a processor according to the present invention;

[0022] Figure 3, a flowchart of the method of selling partially defective processor integrated circuits; and

[0023] Figure 4, a block diagram or portions of a processor according to an alternative embodiment.

DETAILED DESCRIPTION OF THE EMBODIMENTS

[0024] A processor as known in the art receives instructions from cache 99 and 98 through a cache/memory interface 100 (Figure 1) into an instruction fetch unit 102. These instructions are decoded in an instruction decode and dispatch unit 104, which then dispatches them to functional units for execution. Address operations are dispatched to address operation execution units 106, integer operations to integer operation execution units 108, load/store operations to load/store operation execution units 110, and floating point operations to floating point execution units 112.

[0025] Address operation execution units 106, integer operation execution units 108, load/store operation execution units 110, and floating point execution units 112 fetch their operands and store their results in a multiport register file 114. Multiport register file 114 has a large number of ports, for example in one prior art machine it may have twelve read ports, and eight write ports.

[0026] A processor embodying the invention receives instructions from a memory system (not shown), through at least one level of cache 199 (Figure 2) memory, through a cache/memory interface 200 into an instruction fetch unit 202. These instructions are decoded in an instruction decode and dispatch unit 204, which decodes the instructions to determine the types of functional units required to execute each instruction, the registers referenced by each, and other information. Instruction decode and dispatch unit 204 is associated with a branch prediction unit 205. Instruction decode and dispatch unit 204 then checks a status of each functional unit of a required type in a resource status bits 206 register to determine which resources are available and functional.

[0027] In a particular embodiment, the resource status bits 206 register is implemented with programmable memory cells that are programmable at the time of factory test of the integrated circuit on which the processor resides.

[0028] The instruction decode and dispatch unit 204 then dispatches commands to functional units marked usable in resource status bits 206 for execution. These commands are those determined necessary to execute instructions decoded by instruction decode and dispatch unit 204. When sufficient resources are not available to fully dispatch one or more instructions, instruction decode and dispatch unit 204 dispatches those portions for which resources are available and stalls, accepting less than a full group of instructions from instruction fetch unit 202 in the next clock cycle. The remaining portions are then dispatched in a following clock cycle.

[0029] Address operations are dispatched to address operation execution units 214, integer operations to integer operation execution units 208, load/store operations to load/store operation execution units 210, and floating point operations to floating point execution units 212.

[0030] The processor is capable of executing all instructions if one or more functional unit of each type is marked enabled in the resource status bits 206; although performance may be substantially less than with an integrated circuit where all functional units can be enabled. In particular, the processor is capable of executing a mix of integer, floating point, and load/store instructions.

[0031] When a conditional branch instruction is encountered, instruction decode and dispatch unit 204 checks to see if branch prediction unit 205 is marked usable in resource status bits 206. If branch prediction unit 205 is marked usable, it is used to determine the most likely result of the branch condition. The most likely code path is

then speculatively executed. If branch prediction unit 205 is marked as not usable, the branch prediction unit is not used; a predetermined guess is made as to the most likely code path which is then speculatively executed. In a particular embodiment, when branch prediction unit 205 is nonfunctional it is assumed that all conditional branch instructions will be taken, thereby permitting some speculative execution. Since branch prediction accuracy is degraded, the machine can be expected to waste some processor cycles.

[0032] Instruction decode and dispatch unit 204 therefore exhibits degraded performance, but does execute instructions, if at least a minimum subset of functional units are marked usable in resource status bits 206.

[0033] The functional units operate upon information stored in, and store results into, register file 228. Load/store functional units 210 are provided for fetching operands from memory into register file 228, and storing results from register file 228 into memory.

[0034] After fabrication of the integrated circuits containing processors according to Figure 2, each processor of integrated circuits is tested 300 (Figure 3). Testing is as known in the art, where each functional unit is tested, except that a record is kept 302 of defective functional units 208, 210, 212, and 214 identified during testing.

[0035] Once testing 300 is complete, the record of defective functional units is examined. If 304 each processor is fully functional, the integrated circuit is packaged 305 and sold 306 as a full performance unit.

[0036] The record of defective functional units is also checked to determine whether there is a minimum subset 306 of working functional units. If 308 so many defective functional units have been identified that the integrated circuit can not execute all instructions, even at reduced performance, that integrated circuit is scrapped 310.

[0037] The resource status bits 206 are then programmed 312 with information from the record of defective functional units to permanently disable the defective units. Information from the record of defective units is then compared with benchmark results in a performance table 314 to generate a relative performance index, which in an embodiment includes an integer performance subindex and a floating point performance subindex.

[0038] The partially defective integrated circuits are then grouped 316 into bins, or classifications, according to their relative performance index. In a particular embodiment, the grouping is performed such that processors having all integer units 208

working, but either one of two floating point 212 units go into a first bin. Similarly, processors having any one integer unit of several integer units 208 defective with all floating point units 212 working go into a second bin. This simplifies marketing since a smaller number of part numbers and prices are required than if parts were marketed according to which exact subset of functional units are defective.

[0039] The programmed, partially defective, integrated circuits are then stored until market forecasts indicate that there is a market for them.

[0040] When market forecasts indicate that there are likely customers for processors matching their relative performance indexes, the integrated circuits are packaged 318 as known in the art, and sold 320.

[0041] In an alternative embodiment, each functional unit is equipped with built-in self-test (BIST) capability; capable of performing at least some testing of the functional unit at boot time. In this embodiment, upon powerup the resource status bits 206 (figure 2) are written with results of BIST testing. While BIST at powerup is capable of recognizing many permanent failures resulting from processing defects, BIST alone may fail to recognize voltage and temperature related problems in functional units. In this embodiment, where a logical 1 indicates a unit is not available and a logical 0 indicates that it is available for use, resource status bits 206 may be logically or-ed with information from CMOS battery backup memory, or other nonvolatile memory that need not be located on the processor integrated circuit, during boot time. In this embodiment the nonvolatile memory contains information regarding functional units that must be disabled to ensure operation under all voltage and temperature conditions.

[0042] In another alternative embodiment (Figure 4), a processor receives instructions from a memory system (not shown), through first 400 and second 402 level cache through cache/memory interface 404 into an instruction fetch unit 406. These instructions are decoded in an instruction decode and dispatch unit 408, which decodes the instructions to determine the types of functional units required to execute each instruction, the registers referenced by each, and other information. Instruction decode and dispatch unit 408 is associated with a branch prediction unit 410. Instruction decode and dispatch unit 408 then dispatches commands to functional units, including integer units 412, floating point execution units 414, and load/store units 416, for execution. These commands are those determined necessary to execute instructions decoded by instruction decode and dispatch unit 408.

[0043] Each functional unit is associated with a status bit, such as status bits 420 associated with integer/address units 412. Each time a functional unit, such as integer/address units 412, receives a dispatched command, the functional unit polls its status bit and returns an accept signal or a reject signal to the instruction decode and dispatch unit 408. Each unit returns a reject signal if its status bit indicates that it is disabled, and an accept signal if its status bit indicates that it is enabled. Similarly, floating point units 416 have associated status bits 422; and load/store units 416 have associated status bits 426.

[0044] When reject signals are received by the instruction decode and dispatch unit 408, it re-dispatches those portions that were rejected in the following state. It also stalls the instruction stream as required.

[0045] Address and integer operations are dispatched to integer/address operation execution units 412, load/store operations to load/store operation execution units 416, and floating point operations to floating point execution units 414. These units operate on operands stored in a register file 424.

[0046] In this embodiment, best performance is obtained if instruction decode and dispatch unit 408 has the ability to keep track of those functional units that have rejected commands, and preferentially dispatch commands to those units that have not rejected commands.

[0047] In this embodiment, status bits 420, 422, and 426 are set at boot time according to built in self test (BIST) results. These bits are set such that each unit is marked disabled if BIST indicates functional unit failure, and marked enabled if BIST indicates the functional unit is functional.

[0048] While the invention has been particularly shown and described with reference to particular embodiments thereof, it will be understood by those skilled in the art that various other changes in the form and details may be made without departing from the spirit and scope of the invention. It is to be understood that various changes may be made in adapting the invention to different embodiments without departing from the broader inventive concepts disclosed herein and comprehended by the claims that follow.